2002P15893US01

Express Mail No. EV330464164US

Date of Deposit: September 22, 2003

APPLICATION FOR LETTERS PATENT
OF THE UNITED STATES

NAME OF INVENTORS: RICHARD C. SCHAFTLEIN
3 ESTATE COURT
JOHNSON CITY, TN 37604

DANIEL F. MOON
125 DEWEY DUGGER ROAD
JOHNSON CITY, TN 37601

TITLE OF INVENTION: SYSTEM AND METHOD FOR SYNCHRONIZING SYSTEM MODULES

TO WHOM IT MAY CONCERN, THE FOLLOWING IS
A SPECIFICATION OF THE AFORESAID INVENTION

Prepared by

Benjamin M. Rubin, Reg. No. 44,310
170 Wood Avenue South
Iselin, NJ 08830
(732) 321-3113

Docket No. 2002P15893US01
Express Mail No. EV 330464164US

SYSTEM AND METHOD FOR SYNCHRONIZING SYSTEM MODULES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application No.60/413,715, filed on September 26, 2002, incorporated by reference herein, in its entirety.

DESCRIPTION

BACKGROUND OF THE INVENTION

Field of the Invention

The invention generally relates to automatically reassigning an interface card and devices between operating environments, and more particularly, to automatically reassigning interface cards and devices in a programmable logic controller system from a non-deterministic operating environment to a deterministic operating environment.

Background Description

Programmable Logic Controllers (PLCs) typically include a customized piece of hardware that has an instruction set geared for the industrial control industry. This piece of hardware is often referred to as a Hard PLC. The programming language is usually ladder logic or some other language that typically resembles assembly language. The programs (control programs) usually involve the manipulation of memory areas in the PLC that are tied through other specialized pieces of hardware (I/O modules) to machinery such as, for example, switches, pumps, motors, machines, etc. These I/O modules are usually mounted on a rack and can be accessed over a network.

In recent years, the control industry has expanded to take advantage of the power of the Personal Computer (PC). A Soft PLC is a software version of the Hard PLC. Instead of having actual circuitry boards that execute the control program, a software program that runs on the PC executes the program. Additional cards can be inserted into the PC chassis to allow for access to the I/O modules. The Soft PLC will interface with these cards which in turn interface with the I/O modules.

There is another version of the PLC called a Slot PLC. This is a piece of hardware which is in the form of a card that can be inserted into the PC's chassis. The Slot PC behaves like a Hard PLC and has a limited amount of accessibility to other applications on the PC.

Typically, an operating system such as, for example, Windows® (Windows® is a registered trademark of Microsoft Corporation) is used as the environment for the PLC application software to run under and interact with the I/O devices. This environment is a non-deterministic environment and essentially non-real time. However, over time, as the devices or operational requirements of the equipment being controlled changes, the demands on the operating system may become overtaxed and often cannot meet more demanding real-time constraints for new equipment or new requirements for the equipment being controlled. Often this may be simply due to more equipment, for example, on the PLC I/O interfaces vying for more application processing time.

SUMMARY OF THE INVENTION

In an aspect of the invention, a method is provided for reassigning resources in a soft programmable logic controller (PLC). The method includes the steps of selecting an interface in a first operating environment, selecting a virtual slot in a second operating environment for installation of the interface, creating an installation file in the first operating environment for installation of the interface in the second operating environment, and installing the interface in the second operating environment using the installation file to reassign a resource between the first operating environment and the second operating environment.

In another aspect of the invention, a method for automatically reassigning resources in a soft programmable logic controller is provided. The method comprises the steps of identifying a resource to be reassigned from a first processing mode to a second processing mode, and removing the resource from operation in the first processing mode. Further included are the steps of creating an installation file containing information of at least one device driver, assigning the resource for operation in the second operating mode by using installation parameters associated with the first operating mode, and automatically installing the at least one device driver for the resource in the second processing mode using the information from the installation file so that any device in communication with the at least one device driver remains functional.

In another aspect of the invention, a system for reassigning resources in a soft programmable logic controller (PLC) is provided. The system includes a means for selecting an interface in a first operating environment, a means for selecting a virtual slot in a second operating environment for installation of the interface, a means for creating an installation file in the first operating environment for installation of the interface in the second operating environment, and a means for installing the interface in the second operating environment using the installation file to reassign a resource between the first operating environment and the second operating environment.

In another aspect of the invention, a computer program product comprising a computer usable medium having readable program code embodied in the medium is provided. The computer program product includes a first software component to select an interface in a first operating environment, a second software component to select a virtual

slot in a second operating environment for installation of the interface, a third software component to create an installation file in the first operating environment for installation of the interface in the second operating environment, and a fourth software component to install the interface in the second operating environment using the installation file to reassign a resource between the first operating environment and the second operating environment.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of embodiments of the invention with reference to the drawings, in which:

Figure 1 is a diagram of an exemplary PC-based PLC environment;

Figure 2 is an illustration of software architecture of an embodiment of the invention;

Figure 3 is example of a Station Manager's GUI interface screen, according to the invention;

Figure 4 is a flow diagram representing the relationship of a Station Manager Component Configurator with a WinLC Instance Manager;

Figures 5A-5E are diagrams of a user interface created by a Station Manager that provides for configuring a CP card as a submodule; and

Figure 6 is a flow chart showing steps of an embodiment for using the invention.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

This invention is directed to a system and method of extending the operational processing capability of a PC-based PLC by automatically reassigning devices from one operating system configuration to another, e.g., a non-deterministic to a deterministic operating environment. As demands on real-time processing increase on a PC-based PLC, for example, due to addition of controlled devices that interact and control industrial machinery of various types, re-configuration of devices from an essentially non-real-time mode to an essentially real-time mode may provide necessary real time performance to assure adequate response times to and from the controlled devices. The invention provides such real time performance and response times by assuring that the resources are adequately allocated between operating environments that control industrial machinery, for example, reassigning resources in a seamless and automatic manner. The reassignment provides for peak servicing and control of the industrial devices.

Figure 1 is a diagram of an exemplary PC-based PLC environment, generally denoted by reference numeral 1. A PC Station 5 includes an expansion

bus 10 with PC slots 20 that accept board based Slot PLCs, interfaces to Hard PLCs, or interface adaptor cards for soft PLC implementations. Communications 15 for interfacing and controlling devices in a manufacturing environment, or the like, are provided by each board in the PC slots 20, as necessary. The communications 15 may include various network interfaces and topologies, e.g., Ethernet or PROFIBUS-DP (IEC 61158), in communication with controlled devices.

Figure 2 illustrates the software architecture of an embodiment of the invention. In embodiments of the invention, the Microsoft Windows operating system is referenced and used as an example; however, other non-deterministic environments may also be used and are contemplated by the invention. The software architecture of Figure 2 may be implemented on the PC-based environment of Figure 1. Figure 2 is separated into two halves, 100 and 105, by the dashed line 110, representative of two operating system modes. The left hand side, denoted generally by reference numeral 100, is the non-deterministic (non-deterministic herein refers to timing within the WinLC execution of the program; timing can vary from scan to scan in 100, scanning includes a cycle of communication to all devices) operating environment that supports, for example, graphical user interfaces and general-purpose computing.

The right hand side of Figure 2, denoted by reference numeral 105, is deterministic and the scan cycle timing is fixed (non-variable) and provides a real-time execution environment used for processes requiring real-time sensitive

processing. In this example, the Venturcom Real-Time Extensions for Windows (RTX) is referenced and used as an example. The use of the operating system and extensions are provided as examples and illustrative purposes and are not intended as limiting features of the invention.

The Windows environment 100 includes a PC Integration component 115, which defines interfaces such as ISetupSupport and IManageInstance so that PC-based PLC applications may create and manage instances of system components. A portion of the PC Integration component 115 is realized by a Station Manager (SM) component 116. This SM component 116 provides the graphical user interface (GUI) for defining PLC parameters and configurations including the instances of system components. WinLC Setup Manager 120 provides for installation of a software PLC component. This software PLC component is identified to the SM 116 by its component name, which is entered into a registry 130. The registry 130, typically representative of a file, include entries that establish the existence of software PLC components in the system by name and may be called upon by the SM 116.

Table 1 shows an exemplary entry in the WinLC component registry 130.

TABLE 1

Line	Registry Entry
L1	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\StationManager\Components*WinLC*]
L2	"VmdID"=dword:00000002
L3	"ObjID"=dword:0014175c
L4	"CatalogEntry"="WinLC"
L5	"Param"=""
L6	"Status"=dword:00000002
L7	"Flags"=dword:00000002
L8	"Invalid"=dword:00000000
L9	"Databasesize"=dword:00004000

A component name is found in line L1 as "WinLC" of type found in L4 (WinLC). This entry establishes the existence of a component with the name "WinLC." For each instance of WinLC configured, L1 through L9 will exist with a unique name found on L1.

WinLC Instance Manager 125 provides for creating and managing one or more instances of the software PLC 180 components as directed by the Station Manger 120. These instances may be implemented as one or more software applications.

Still referring to Figure 2, a software PLC 180 comprises two major components: i) a CPU proxy 135, which manages user interaction with the software PLC; and, ii) an execution engine, comprising a CPU Stub 145 and a Virtual CPU 150. The CPU Proxy 135 runs on the non-deterministic 100 side of the system while the execution engine runs on the real-time side 105 of the system. These two halves of the software PLC application communicate bi-

directionally through a shared memory arrangement 140. The CPU proxy 135 and its associated execution engine are instantiated for every instance of the software PLC 180.

In the invention, the CPU proxy 135 coordinates the control and communication of events to and from the non-deterministic100 side and the real-time side 105. For example, the CPU Proxy 135 might command the execution engine to start or stop execution or the execution engine might notify the CPU proxy 135 of a fault condition to be displayed on an operator panel. The communication across the non-determinist and real-time boundary 110 may be accomplished through the shared memory arrangement 140, or similar arrangement. The CPU stub 145 provides the linkages to the virtual CPU instance 150 (also known as V7 CPU) of the real-time side 105.

By using components such as 135 ,140, 145, the invention is capable of reassigning control from the non-deterministic side 100 to the deterministic side 105 (and like version) to ensure proper allocation and efficient use of resources in the soft PLC environment. The reassignment may be performed based on various reasons such as slowing of response time to devices associated with the non-deterministic side, user action, segregation of certain devices by environment, critical demands of certain device operations, prioritization of devices, etc.

The real-time operating system on the real-time side 105 provides multiple threads of execution to enable more real-time oriented processing since each thread may receive a unique priority and may receive higher priority over the non-

deterministic side 100 processing, as necessary. By moving time-sensitive portions of the application from the non-deterministic side 100 to the real-time 105 side, real-time demands may be better achieved and achieved more dependably. Reassignment of system resources, e.g., device drivers 165 with their interface cards, from the non-deterministic side 100 to the real-time side 105 is performed automatically with minimal user intervention (shown in Fig. 6).

Figure 3 is an example of a SM's 116 GUI interface screen 170 for assigning or modifying an entry in the component registry 130. As illustrated, the second entry 175 shows that a component with the name WinLC has been created. By clicking on this entry, further parameters and configuration of WinLC may be made.

Table 2 shows a WinLC registry entry associated with component WinLC

TABLE 2

L1	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC]
L2	"Product_Version"=""
L3	"Product_Key"="_Basis"
L4	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11]
L5	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11\CP5611(PROFIBUS)]
L6	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11\CurrentVersion]
L7	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11\DeviceID]
L8	"*SIE9020"="YES"
L9	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11\General]
L10	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\DP5x11\General\Setup]
L11	"InstLangs"="B"
L12	"InstComps"=""
L13	"BaseLangs"="B"
L14	"BaseComps"=""
L15	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\Installed Instance]
L16	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\Installed Instance\WINLC0]
L17	"UserNameForInstance"="WinLC"
L18	"VmdID"=dword:00000002
L19	"StartAtPCBoot"=dword:00000000

```
"Instance Type"="_BASIS"
"pwd"=""
"SecurityLevel"=dword:00000000
"Active File Path"="C:\\SIEMENS\\WINAC\\WinLC\\WINLC0.waf"
"Background Processing Priority"=dword:00000003
"Foreground Processing Priority"=dword:00000005
"I/O Type"="DP"
"Interrupt Priority"=dword:00000006
"Min Scan Sleep(ms)"=dword:0000000a
"OB Execute Priority"=dword:00000004
"OB Sleep Interval(us)"=dword:000003e8
"OB Wake Interval(us)"=dword:00002328
"Process Priority"=dword:00000003
"RemoteHost"=""
"Startup Mode"="MANUAL"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\Installed Instance\WINLC0\Sub-Modules]
"IF1"="CP5611(PROFIBUS)"
"IF2"=""
"IF3"=""
"IF4"=""

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis]
"Cyclic Interrupt OB Overrun Depth"=dword:0000001e
"Display Update(ms)"=dword:00000064
"InstanceRegPath"="SIEMENS\\WINLC\\Installed Instance"
"K-Bus Driver Path"="s7wltonx.dll"
"ProxyCPU Name"="s7wlcapx.exe"
"VirtualCPU Name"="s7wlcvmx.exe"
"ShowDebugFaceplateInRel"="off"
"Version"="4.0"
"Path"="C:\\SIEMENS\\WINAC"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\Addons]

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\Addons\ExtensionDLLs]
"ODK"="s7odkvmx.dll"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\CurrentVersion]

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\Drivers]

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\Drivers\CP5611]
"CP Control Path"="s7wlcdpx.dll"
"DP Library Path"="s7wlcdpx.dll"
"TPI Library Path"="s7wltonx.dll"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\Drivers\CP5613_5614]
"CP Control Path"="EpDiagEx.dll"
"DP Library Path"="DPLib_5613.dll"
"TPI Library Path"="s7wltonx.dll"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\General]

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\General\InstLanguages]
"Applications"="ENGLISH,GERMAN,FRENCH,"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\General\Languages]
"Current"="ENGLISH"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC\_Basis\General\Paths]
"Product_Dir"="C:\\SIEMENS\\WINAC"
"Product_Bin_SDir"="WinLC"
"Product_Short_Dir"="C:\\SIEMENS\\WINAC"
"Example_Dir"="C:\\SIEMENS\\WINAC\\Examples"
```

	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC_Basis\General\RelVersion]
L74	"ReleaseVersion"="4.0"
L75	
	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC_Basis\General\Setup]
L76	"InstLangs"="BAC"
L77	"InstComps"=""
L78	"BaseLangs"="BAC"
L79	"BaseComps"=""
L80	"FolderName"="PC Based Control"
L81	"Restart"="NO"
L82	[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\WINLC_Basis\General\UserInfo]
L83	"Name"="Siemens"
L84	"Company"="Siemens"
L85	

Table 2 defines many attributes and parameters for the internal instance named "WinLC0" as defined in line L16. Line L15 through L34 defines the WinLC instance. Line L17 defines the user recognizable name "WinLC". This is the same name as defined in Table 1, line L4. An entry in Table 1, L1 through L9, corresponds to an entry in Table 2, L16 through L34. Lines L16 through L40 are unique to each WinLC instance configured in the SM 116.L4 through L14 define initialization parameters for the CP Card and the language used by any dialog or message that may appear to the user. L40 through L85 define locations of program files, initialization parameters, default operating modes, and language used by any dialog or message that may appear to the user for each WinLC instance.

Table 3 shows the instance specific registry keys that the WinLC Instance Manger 125 creates, modifies and deletes as necessary in response to a SM 116 directive. A unique instance specific registry key is created using the names defined in Table 3 whenever a WinLC instance is added to the system. All WinLC instances of all WinLC product types are managed by the WinLC Instance Manger 125 and are

defined under the "Installed Instance" registry location (L15) as defined in Table 2 above.

TABLE 3

• *WinLC*
• *WinLC1*
• *WinLC2*
• ...

Figure 4 is a flow diagram representing the relationship of a SM Component Configurator with a WinLC Instance Manger 125. As depicted in Figure 4, the SM 116 calls the IMangeInstance interface of the WinLC Instance Manger 125 for any additions, modifications, and/or deletions of WinLC instances. The WinLC Instance Manager 125 can create, modify, and/or delete the instance specific registry key in response to a SM Component Configurator. When a WinLC instance is added to the PC Station 5, a unique instance specific registry key is created using the names defined in Table 3.

Other entries of the WinLC Registry entry of Table 2 are created by the WinLC Instance Manager. For example, the WinLC Instance Manager creates the "StartAtPCBoot" (L19) (defaulted to be false), the "UserNameForInstance" (L17) (from the user defined name that is shown in the name column on the SM Component Configuration Editor of Figure 3), the "VmdID" (L18) (the index value on the SM

Component Configurator), and the "Instance Type" (L20). The remaining entries under the "Installed Instance" key are created the first time the WinLC Instance is run.

The WinLC Instance Manager 125 manages the instances of all WinLC product types on a single PC Station 5. In embodiments, all WinLC Instances are located under the "Installed Instance" registry location as defined above in Table 2. The WinLC Instance Manager may implement the IManageInstance interface of the SM 116. The SM 116 calls the IManageInstance interface of the WinLC Instance Manager 125 for any additions, modifications, and/or deletions of WinLC instances on the PC Station 5.

The WinLC Instance Manager 125 also creates a menu entry in the PC Start Menu for each configured WinLC Instance (Start > Simatic > PC Based Control > *WinLC*, where *WinLC* is the user defined name for the WinLC Instance. It also creates an icon on the PC Desktop for each configured WinLC Instance (desktop icon named WinLC, where WinLC is the user defined name for the WinLC Instance). The menu entries and desktop icons are created when a WinLC Instance is added, they are modified when a WinLC Instance is edited, and they are removed when a WinLC Instance is deleted from the SM 116. The menu entries may appear in the GUI of Figures 5A-5E.

The WinLC user-program (control software) is created and maintained using an engineering system (for example, the Siemens STEP 7 product, which is the graphical tool for programming the WinLC). The engineering system communicates with WinLC using a communication protocol that provides for program upload, download, and debug, control variable monitoring, alarming, etc. (this protocol may include, for example, the Siemens Multi-Point Interface, or

MPI) Additionally, a WinLC instance communicates with its distributed I/O over a process field-bus network, e.g., PROFIBUS-DP. Both of these networking functions are implemented using communications processor (CP) cards 20 which plug into the expansion bus (backplane) 10 of the PC 5.

The association between a given CP card and its WinLC instance is made when the user places the CP card into the PC Station using the SM application. A CP card may be inserted into the PC Station in one of two logically-distinct ways: as a component of the PC Station's virtual backplane or as a submodule of the WinLC instance. The user operation which accomplishes the insertion is performed using graphical interfaces and is similar to installing a module in a rack of a hard PLC.

After a CP card has been configured as a component of the PC Station 5, it can be used for MPI communications, for example, to download a program from the engineering system to the WinLC instance. The WinLC instance accesses CP cards that are installed as components of the PC Station's 5 virtual backplane and slots using software protocols that emulate the backplane of a hard PLC, e.g., a Siemens S7 400. These software protocols operate on the essentially non-deterministic side 100 of the environment. Consequently, they are accessed through the CPU Proxy 135 and CPU Stub 145.

A CP card that has been configured as a submodule of a WinLC instance can be used for MPI communication as well as for accessing the instance's I/O. These CP cards are directly accessed by the WinLC instance's execution engine

(i.e., access does not use backplane emulation and does not involve the CPU Proxy or the SPC Stub.) This means, for example, that the card is controlled in the real-time environment 105 and that this control is deterministic.

Figures 5A-5E show graphical user interfaces, generally denoted by reference numeral 180 and may be created by the SM 116 that provides for configuring a CP card as a submodule of a WinLC instance. Figure 6 is a flow chart of steps of an embodiment for using the invention that begins at step 210 and is discussed concurrently with Figures 5A-5E. Figure 6 may equally represent a high-level block diagram of components of the invention implementing the steps thereof. The steps of Figure 6 may be implemented on computer program code in combination with the appropriate hardware. This computer program code may be stored on storage media such as a diskette, hard disk, CD-ROM, DVD-ROM or tape, as well as a memory storage device or collection of memory storage devices such as read-only memory (ROM) or random access memory (RAM). Additionally, the computer program code can be transferred to a workstation over the Internet or some other type of network. Figure 6 may be substantially implemented via the GUI of Figures 5A-5E.

To configure a CP card as a submodule of a WinLC instance, at step 220, the user requests the WinLC instance's properties dialog as follows: select the WinLC instance (e.g., at index 2 of Fig. 5A); select "Edit" from either the right-click menu or from the Edit button; select the "Properties" button from the "Edit Component" dialog (Fig. 5B). At step 230, the "WinLC Properties" dialog will be displayed (Fig. 5C).

From the WinLC Properties dialog of Fig. 5C and at step 240, the user then selects an empty interface slot (IF), e.g., index IF1-IF4, and the requests the "Add" function from either the right-click menu or from the Add button. At step 250, the "Station Manager CP Modules" dialog is displayed as shown in Fig. 5D. This dialog lists the CP cards (e.g., CP5613) that can be moved from the non-deterministic side 100 (e.g., from the SM's virtual backplane) to the real-time side 105 as a submodule of the WinLC instance.

At step 260, selecting a CP card (e.g., index 181) from this list is executed, followed, at step 270, with a check to insure that the submodule name is unique. Clicking the "OK" button causes, at step 280, initiation of a move of the card from the SM's virtual backplane position to the WinLC instance at the selected IF position. The result of the move is shown in Fig. 5E, which shows the CP5613 to reside at interface slot IF1. The sequence above may be repeated as necessary for each card to be moved as a submodule of WinLC.

An aspect of this invention which significantly simplifies the user's task includes the operational steps of moving the CP card from the SM's virtual backplane to the WinLC instance's I/F slot (submodule position). This simple point-and-click user operation hides from a user a sequence of operations as shown in steps 280 through 400 of the flow chart in Figure 6.

Prior to the user clicking OK in Fig. 5D, at step 280, the CP card exists as a device in the non-deterministic operating system environment. As such, it has a non-deterministic (e.g., Windows Driver Model) device driver and is only available for use by

non-deterministic applications, e.g., for use as an MPI (Multi-Point Interface) communication controller. Following the user clicking OK, the CP card is no longer available in the non-deterministic operating system environment. Instead, it is installed as a real-time device with exclusive access by the WinLC instance's execution engine.

The move may be accomplished using the following steps. At step 270, the WinLC Instance Manager 125 validates that the user-assigned name for the card is unique. If not, two actions are possible: the system can generate a unique name; or the user can be prompted to supply a unique name. At step 280, the WinLC Instance Manager 125 invokes the SM 116, instructing it to remove the card from the SM's virtual backplane. At step 290, the WinLC Instance Manager 125 creates an installation file that allows the card to be installed for real-time use. This file may be used to override the normal (non-deterministic) installation of the card, which is specified by the installation file of the card's default driver. At step 300, the installation parameters of the card, as currently installed, are obtained from the operating system. At step 310, the current installation parameters are modified to change the card's installation file from its current setting (referencing the default driver) to reference the installation file created at step 290.

At step 320, a list of compatible drivers satisfying the modified installation parameters is generated and supplied to the Windows Operating System (for example). Due to the manner in which the installation file is created at step 290, this list includes only the real-time driver. At step 330, the operating system is queried to determine whether the real-time driver has been installed. If not, at step 340, the real-time driver will be installed before proceeding. At step 350, the operating system is instructed to

uninstall the card. This breaks the card's relationship to the current (non-deterministic) driver and makes the card unavailable for use by applications residing on the non-deterministic side of the architecture 100. At step 360, the operating system is instructed to reinstall the card using the real-time driver. This operation assigns control of the card to the real-time side of the architecture 105, limiting its used to the WinLC instance's execution engine.

At step 370, the installation file created at step 290 is removed from the system. This prevents the operating system from silently installing a subsequently inserted card of the same type to the real-time driver. In other words, all cards typically start out their life assigned to their non-deterministic driver. Explicit user action is required to indicate that a card is to be assigned to the real-time driver and reside in the real-time environment. At step 380, the real-time operating system 105 is instructed, as necessary, to allow the card's interrupt line to be shared with other real-time cards. Since interrupt lines are generally connected to more than one expansion slot (card), this provides a higher potential that multiple cards may be moved to the real-time environment.

At step 390, the WinLC instance's submodule information is updated in the SM's component registry and in the WinLC installed instance registry. The card is now listed as a submodule of the instance. At step 400, control is returned to the User, e.g., as shown in Fig. 5E. The real time environment provides assurance of guaranteed fixed scan cycles with known response time capability to and from devices on the assigned virtual and real ports on the backplane/expansion bus 10. While the steps of Figure 6 illustrate moving resources from a non-deterministic environment to a

deterministic environment, one of ordinary skill in the art would recognize that movement of resources in the reverse direction, i.e., from a deterministic environment to a non-deterministic environment is also possible.

While the invention has been described in terms of embodiments, those skilled in the art will recognize that the invention can be practiced with modifications and in the spirit and scope of the appended claims.